MathWorks
AUTOMOTIVE CONFERENCE 2016

9월 2일 ~ 서울
Establishing Development Process and Automation

MathWorks
Consulting Services Group
Today’s Agenda

- Verification Activities in MBD
- Automation of Model Verification Activities
  - Model Testing with Simulink Test
- Build Automation of Generated Code and Legacy Code Integration
- Automation of Polyspace Code Verification
  - Coding Standard Checks
  - Formal Verification (Absence of Run-Time Error)
- Questions and Answers
Verification Activities in MBD

- **Model Verification**
  - Design Review
    - Requirement Traceability
    - Report Generation
  - Static Analysis
    - Model Standards Checking
    - Design Error Detection
    - Prove Design Correctness
  - Dynamic Test
    - Simulation-based Functional Test
    - Coverage Analysis
    - Simulink Test
Verification Activities in MBD (cont’d)

- **Code Generation and Verification**
  - **Code Review**
    - Code Generation Report
  - **Static Analysis**
    - Code Metrics and Coding Rule Checking
    - Formal Verification (Abstract Interpretation)
  - **Equivalence Test**
    - SIL(Software-In-the-Loop) and PIL(Processor-In-the-Loop)
Motivation

**New Users / Early stage MBD adoption**
- Map and Prioritize Processes
- Model Architecture Establishment and Review
- Tool implementation support

**Advanced Users / Continuous verification**
- Automate various verification activities
- Continuous model and code review reports
**Model-Based Design for ISO 26262 ASIL D Safety Functions**

- **Observations**
  - Many of the advanced analysis and design techniques called out by the standard are manually intensive to perform using traditional methods
    - e.g. Range checks of input/output data, Diverse SW Design, Prototype generation
  - Model-Based Design supports many of methods called out by the standard and provides automation to further reduce the manual effort

---

### Process Steps

<table>
<thead>
<tr>
<th>Methods</th>
<th>Process Steps</th>
<th>1a</th>
<th>1b</th>
<th>1c</th>
<th>1d</th>
<th>1e</th>
<th>1f</th>
<th>1g</th>
<th>1h</th>
<th>1i</th>
<th>1j</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
</tbody>
</table>

---

*Table 1 Table 2 Table 3 Table 4 Table 5 Table 6 Table 7 Table 8 Table 9 Table 10 Table 11 Table 12 Table 13 Table 14 Table 15 Table 16*
Adopting Capabilities to Optimize Model-Based Design for ISO 26262
CI Tool Integration with MBD

- CI Tool (Jenkins) Integration for Model Based Verification Activities
  - Automation of Modeling Testing
  - Build Automation for Code Integration (Generated Code and Legacy Code)
  - Automation of Polyspace Code Verification
    - Coding Standard Checks
    - Formal Verification (Absence of Run-Time Error)
Integration with CI Tools: Environment and Workflow

1. Regular (ex. 10PM every day) update developers’ model from SVN repository
2. Automatically trigger MATLAB application
3. Regular Test Reports
Automation of Model Verification
Motivation
: Model Verification (Simulink Test)

- Continuous Model Verification Test
  - Tedious repetitive tests
  - Pre-defined test scenarios
  - Simulation vs expected behaviors by test scenario
  - Regression test
  - Result analysis
  - Simulink Test Report
  - EARLY detected potential bugs
Automated Model Testing with Jenkins

DEMO
Continuous Integration Builds
Motivation
: Daily build / Continuous integration builds

- The practice of each day doing software build of the latest version
  - Compiled to ensure all required dependencies are present
  - *Build* is the results of compiling and linking
  - Particularly necessary in large organizations
  - New bugs that shows up as a results of work done within the last day
Motivation
: Daily build / Continuous integration builds

Basic Software
- Hand Written Code

Application Software
- Model Generated Code

H/W & Device Driver

Interface integration (Input, Output and Parameters)
Early Detection through continuous integration builds
Binary and a2l file generation
Build Automation with Jenkins

DEMO
Automation of Polyspace Code Verification
Motivation:

: Code Verification (Polyspace)

- Continuous Code Verification Test
  - *Pre-defined* compliance with Coding Standards
  - Reduce Test Cycles by Fixing Bug early
    - Provide detailed information on what caused the defect
    - Tracing Code Analysis Results to Simulink Models
  - Code Metrics and Monitor Software Quality
    - Progress of software quality
Automated Polyspace Code Verification with Jenkins

DEMO
Conclusion

Productivity Improvement Dynamics

→ Speed Return on Investment in technology
Q&A